NAME

hcreate, **hcreate_r**, **hdestroy**, **hdestroy_r**, **hsearch**, **hsearch_r** - manage hash search table

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <search.h>

int
hcreate(*size_t nel*);

int
hcreate_r(*size_t nel*, *struct hsearch_data *table*);

void
hdestroy(*void*);

void
hdestroy_r(*struct hsearch_data *table*);

*ENTRY **
hsearch(*ENTRY item*, *ACTION action*);

int
hsearch_r(*ENTRY item*, *ACTION action*, *ENTRY ** itemp*, *struct hsearch_data *table*);

DESCRIPTION

The **hcreate**(), **hcreate_r**(), **hdestroy**(), **hdestroy_r**() **hsearch**(), and **hsearch_r**() functions manage hash search tables.

The **hcreate**() function allocates sufficient space for the table, and the application should ensure it is called before **hsearch**() is used. The *nel* argument is an estimate of the maximum number of entries that the table should contain. As this implementation resizes the hash table dynamically, this argument is ignored.

The **hdestroy**() function disposes of the search table, and may be followed by another call to **hcreate**(). After the call to **hdestroy**(), the data can no longer be considered accessible. The **hdestroy**() function calls free(3) for each comparison key in the search table but not the data item associated with the key.

The **hsearch**() function is a hash-table search routine. It returns a pointer into a hash table indicating the location at which an entry can be found. The *item* argument is a structure of type *ENTRY* (defined in the <*search.h*> header) that contains two pointers: *item.key* points to the comparison key (a *char* *), and *item.data* (a *void* *) points to any other data to be associated with that key. The comparison function used by **hsearch**() is strcmp(3). The *action* argument is a member of an enumeration type *ACTION* indicating the disposition of the entry if it cannot be found in the table. ENTER indicates that the *item* should be inserted in the table at an appropriate point. FIND indicates that no entry should be made. Unsuccessful resolution is indicated by the return of a NULL pointer.

The comparison key (passed to **hsearch**() as *item.key*) must be allocated using malloc(3) if *action* is ENTER and **hdestroy**() is called.

The **hcreate_r**(), **hdestroy_r**(), and **hsearch_r**() functions are re-entrant versions of the above functions that can operate on a table supplied by the user. The **hsearch_r**() function returns 0 if the action is ENTER and the element cannot be created, 1 otherwise. If the element exists or can be created, it will be placed in *itemp*, otherwise *itemp* will be set to NULL.

RETURN VALUES

The **hcreate**() and **hcreate_r**() functions return 0 if the table creation failed and the global variable *errno* is set to indicate the error; otherwise, a non-zero value is returned.

The **hdestroy**() and **hdestroy_r**() functions return no value.

The **hsearch**() and **hsearch_r**() functions return a NULL pointer if either the *action* is FIND and the *item* could not be found or the *action* is ENTER and the table is full.

EXAMPLES

The following example reads in strings followed by two numbers and stores them in a hash table, discarding duplicates. It then reads in strings and finds the matching entry in the hash table and prints it out.

```
#include <stdio.h>
#include <search.h>
#include <string.h>
#include <stdlib.h>

struct info {                   /* This is the info stored in the table */
        int age, room;          /* other than the key. */
};
```

```
#define NUM_EMPL        5000    /* # of elements in search table. */

int
main(void)
{
        char str[BUFSIZ]; /* Space to read string */
        struct info info_space[NUM_EMPL]; /* Space to store employee info. */
        struct info *info_ptr = info_space; /* Next space in info_space. */
        ENTRY item;
        ENTRY *found_item; /* Name to look for in table. */
        char name_to_find[30];
        int i = 0;

        /* Create table; no error checking is performed. */
        (void) hcreate(NUM_EMPL);

        while (scanf("%s%d%d", str, &info_ptr->age,
           &info_ptr->room) != EOF && i++ < NUM_EMPL) {
                /* Put information in structure, and structure in item. */
                item.key = strdup(str);
                item.data = info_ptr;
                info_ptr++;
                /* Put item into table. */
                (void) hsearch(item, ENTER);
        }

        /* Access table. */
        item.key = name_to_find;
        while (scanf("%s", item.key) != EOF) {
                if ((found_item = hsearch(item, FIND)) != NULL) {
                        /* If item is in the table. */
                        (void)printf("found %s, age = %d, room = %d\n",
                            found_item->key,
                            ((struct info *)found_item->data)->age,
                            ((struct info *)found_item->data)->room);
                } else
                        (void)printf("no such employee %s\n", name_to_find);
        }
        hdestroy();
        return 0;
```

```
}
```

ERRORS

The **hcreate**(), **hcreate_r**(), **hsearch**(), and **hsearch_r**() functions will fail if:

[ENOMEM] Insufficient memory is available.

The **hsearch**() and **hsearch_r**() functions will also fail if the action is FIND and the element is not found:

[ESRCH] The *item* given is not found.

SEE ALSO

bsearch(3), lsearch(3), malloc(3), strcmp(3), tsearch(3)

STANDARDS

The **hcreate**(), **hdestroy**(), and **hsearch**() functions conform to X/Open Portability Guide Issue 4, Version 2 ("XPG4.2").

HISTORY

The **hcreate**(), **hdestroy**(), and **hsearch**() functions first appeared in AT&T System V UNIX. The **hcreate_r**(), **hdestroy_r**() and **hsearch_r**() functions are GNU extensions.

BUGS

The original, non-GNU interface permits the use of only one hash table at a time.